



Московский Государственный Университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

**Отчет по заданию №2 в рамках курса**  
**«Суперкомпьютерное моделирование и технологии»**  
**Численное интегрирование многомерных функций**  
**методом Монте-Карло**

**Выполнил:**

студент группы 608

Канзепаров Денис Ринатович

Вариант № 9

Москва, 2022

# Содержание

1. Математическая постановка задачи . . . . .	2
2. Численный метод решения . . . . .	2
3. Нахождение точного значения интеграла аналитически . . . . .	3
4. Исследование масштабируемости программы на системе Polus. . . . .	4
5. Краткое описание прогораммной реализации . . . . .	4
6. Выводы . . . . .	6

# 1. Математическая постановка задачи

Функция  $f(x, y, z)$  – непрерывна в ограниченной замкнутой области  $G \subset \mathbb{R}^3$ . Требуется вычислить определенный интеграл:

$$I = \int \int \int_G f(x, y, z) dx dy dz.$$

Интегрируемая функция:

$$f(x, y, z) = xy^2z^3,$$

область  $G$  ограничена поверхностями  $z = xy$ ,  $y = x$ ,  $x = 1$ ,  $z = 0$ .

# 2. Численный метод решения

Пусть область  $G$  ограничена параллелепипедом  $\Pi$ :

$$\begin{cases} a_1 \leq x \leq b_1, \\ a_2 \leq x \leq b_2, \\ a_3 \leq x \leq b_3. \end{cases}$$

Рассмотрим функцию  $F(x, y, z)$ :

$$F(x, y, z) = \begin{cases} f(x, y, z), & (x, y, z) \in G \\ 0, & (x, y, z) \notin G. \end{cases}$$

Преобразуем искомый интеграл:

$$I = \int \int \int_G f(x, y, z) dx dy dz = \int \int \int_{\Pi} F(x, y, z) dx dy dz.$$

Пусть  $p_1(x_1, y_1, z_1), p_2(x_2, y_2, z_2), \dots$  – случайные точки, равномерно распределенные в  $\Pi$ . Возьмем  $n$  таких случайных точек. В качестве приближенного значения интеграла предлагается использовать выражение:

$$I \approx |\Pi| \cdot \frac{1}{n} \sum_{i=1}^n F(p_i), \quad (1)$$

где  $|\Pi|$  – объем параллелепипеда  $\Pi$ .  $|\Pi| = (b_1 - a_1)(b_2 - a_2)(b_3 - a_3)$ .

### 3. Нахождение точного значения интеграла аналитически

Вычислим интеграл аналитически:

$$\begin{aligned} I &= \int_0^1 \int_0^x \int_0^{xy} xy^2 z^3 dx dy dz = \int_0^1 \int_0^x xy^2 \frac{(xy)^4}{4} dx dy = \frac{1}{4} \int_0^1 \int_0^x x^5 y^6 dx dy = \frac{1}{4} \int_0^1 x^5 \frac{x^7}{7} dx = \\ &= \frac{1}{28} \int_0^1 x^{12} dx = \frac{1}{28} \frac{1}{13} = 0.00(274725) \approx 0.002747252747 \end{aligned}$$

Далее в качестве точного решения данного интеграла будем использовать значение 0.002747252747.

### 4. Исследование масштабируемости программы на системе Polus.

Для корректной работы парадигмы «мастер-рабочие» будем фиксировать выборку для 2, 4, 16, 32 MPI-процессов. Фиксировать будем путем фиксации зерна генератора (seed). В качестве значений для seed'a были выбраны значения 2, 20, 200, 2000. По результатам работы программы мы видим, что для каждого значения seed'a ошибка постоянна при постоянном количестве точек соответственно (в зависимости от заданной точности). Также в качестве одного из вариантов ускорения работы был рассмотрен вариант деления выборки на партии (помимо деления на количество процессов). И рассылка выборки «мастером» этих батчей по «рабочим».

### 5. Краткое описание пропрограммной реализации

Реализуем парадигму «мастер-рабочие»: один из процессов («мастер») генерирует случайные точки и передаёт каждому из остальных процессов («рабочих») отдельный, предназначенный для него, набор сгенерированных случайных точек. Все процессы-рабочие вычисляют свою часть суммы по формуле (1). Затем вычисляется общая сумма с помощью операции редукции.

«Мастер»: проверяет условие сходимости, если оно не выполнено, то продолжает работу, а именно, увеличивает число создаваемых точек в 2 раза, определяет новое зерно для вычисления случайных координат точек, создает массив с координатами точек. Далее начинает рассылать данный массив по «рабочим».

«Рабочий»: проверяет условие сходимости, если оно не выполнено, то продолжает работу, а именно, принимает массив случайных точек, суммирует значения функции в точках, которые лежат в области  $G$ , вычисляет значение интеграла по формуле (1). Далее высылает полученный результат на «мастер».

«Мастер»: складывает полученные с «рабочих» результаты, проверяет верность условия сходимости и рассылает по «рабочим» значение переменной, отвечающей за прекращение вычислений, если достигнута точность. Если точность достигнута, то выводится результат работы программы. Иначе делает еще одну итерацию.

## 6. Выводы

В данной работе была реализована парадигма «мастер-рабочие» для вычисления кратного интеграла численным методом Монте-Карло. По результатам, полученными выполнением программной реализации, можно сказать, что задача масштабируема, но в зависимости от настраиваемых параметров (гиперпараметров). Одним из которых в первую очередь является зерно генератора. Мы видим, что при реализации мы не получили возможного кратного ускорения, это связано с тем, что в парадигме «мастер-рабочие» при увеличении рабочих увеличивается время пересылки.

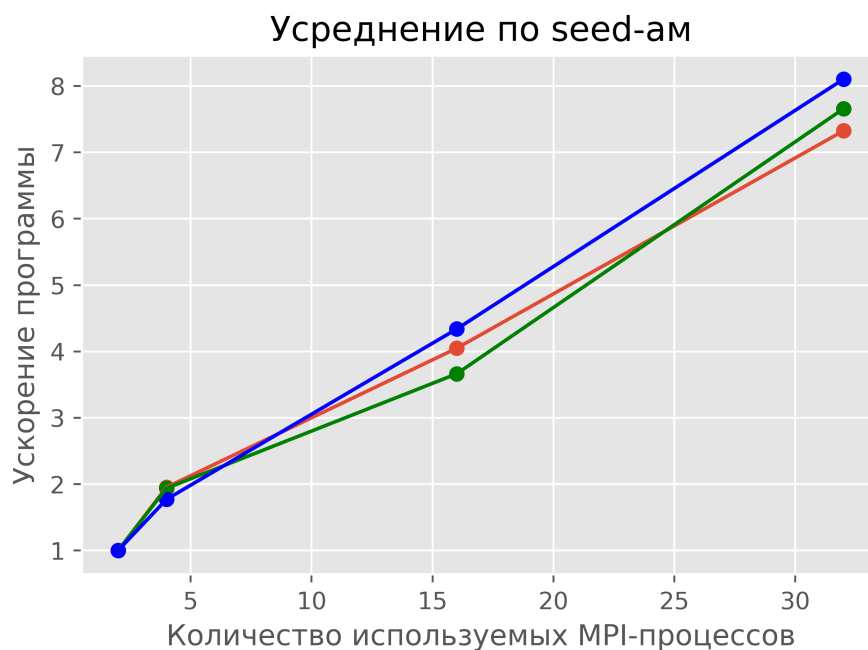


Рис. 6.1. Графики зависимости ускорения программы от числа используемых MPI-процессов со значениями усредненными по seed'ам

Таблица 1. Таблица с результатами расчётов для **seed = 2**

Точность $\varepsilon$	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	2	0.0129639100	1	0.0000271810
	4	0.0079651151	1.6275860219	0.0000271810
	16	0.0022330770	5.8054021424	0.0000271810
	32	0.0013891429	9.3323084327	0.0000271810
$5.0 \cdot 10^{-6}$	2	0.0767475457	1	0.0000037189
	4	0.0393421291	1.9507725549	0.0000037189
	16	0.0209214501	3.6683664532	0.0000037189
	32	0.0079084291	9.7045247203	0.0000037189
$1.5 \cdot 10^{-6}$	2	0.6942877981	1	0.0000009119
	4	0.3703793331	1.8745316923	0.0000009119
	16	0.1480963470	4.6880818613	0.0000009119
	32	0.1247988479	5.5632548679	0.0000009119

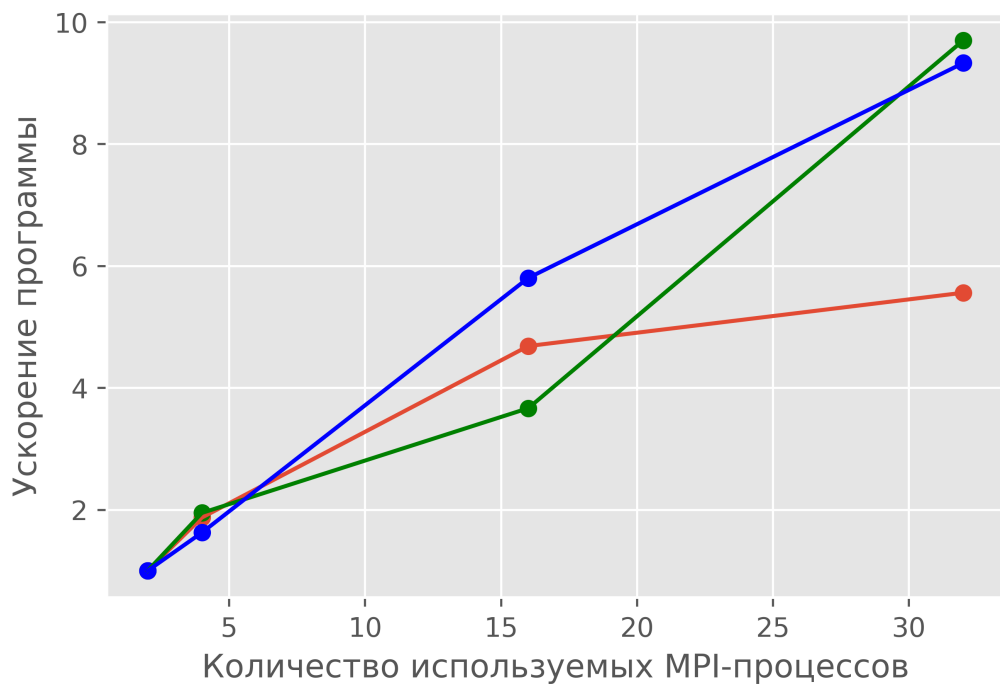


Рис. 6.2. Графики зависимости ускорения программы от числа используемых MPI-процессов **seed = 2**

Таблица 2. Таблица с результатами расчётов для **seed = 20**

Точность $\varepsilon$	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	2	0.0237541629	1	0.0000157098
	4	0.0136748330	1.7370715167	0.0000157098
	16	0.0073593007	3.2277744677	0.0000157098
	32	0.0035384192	6.7132133185	0.0000157098
$5.0 \cdot 10^{-6}$	2	0.0344767939	1	0.0000028539
	4	0.0191537743	1.8000000083	0.0000028539
	16	0.0106409857	3.2400000218	0.0000028539
	32	0.0056005187	6.1560001397	0.0000028539
$1.5 \cdot 10^{-6}$	2	0.6767938490	1	0.0000005337
	4	0.3405903262	1.9871200000	0.0000005337
	16	0.1892168478	3.5768160016	0.0000005337
	32	0.0901032608	7.5113136083	0.0000005337

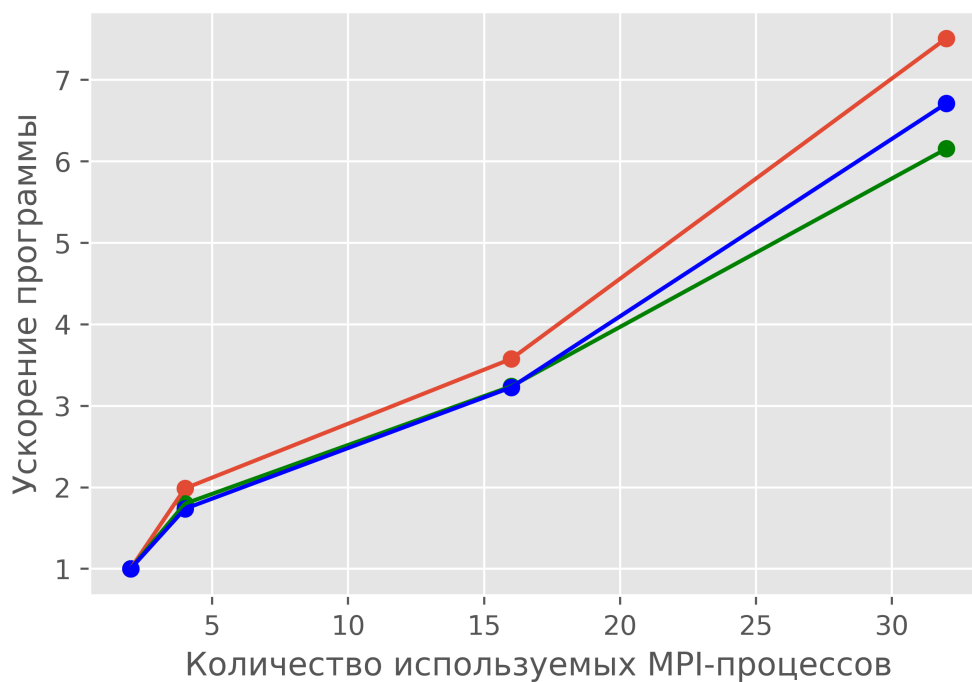


Рис. 6.3. Графики зависимости ускорения программы от числа используемых MPI-процессов **seed = 20**

Таблица 3. Таблица с результатами расчётов для **seed = 200**

Точность $\varepsilon$	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	2	0.0120780700	1	0.0000058787
	4	0.0063942753	1.8888880183	0.0000058787
	16	0.0029064887	4.1555537442	0.0000058787
	32	0.0012636907	9.5577739078	0.0000058787
$5.0 \cdot 10^{-6}$	2	1.4561738340	1	0.0000028539
	4	0.6823059294	2.1341948988	0.0000028539
	16	0.3529457491	4.1257724103	0.0000028539
	32	0.2190974159	6.6462391992	0.0000028539
$5.0 \cdot 10^{-6}$	2	2.9588420690	1	0.0000010679
	4	1.3229571294	2.2365366218	0.0000010679
	16	0.6391130261	4.6296068898	0.0000010679
	32	0.3291077191	8.9904973274	0.0000010679

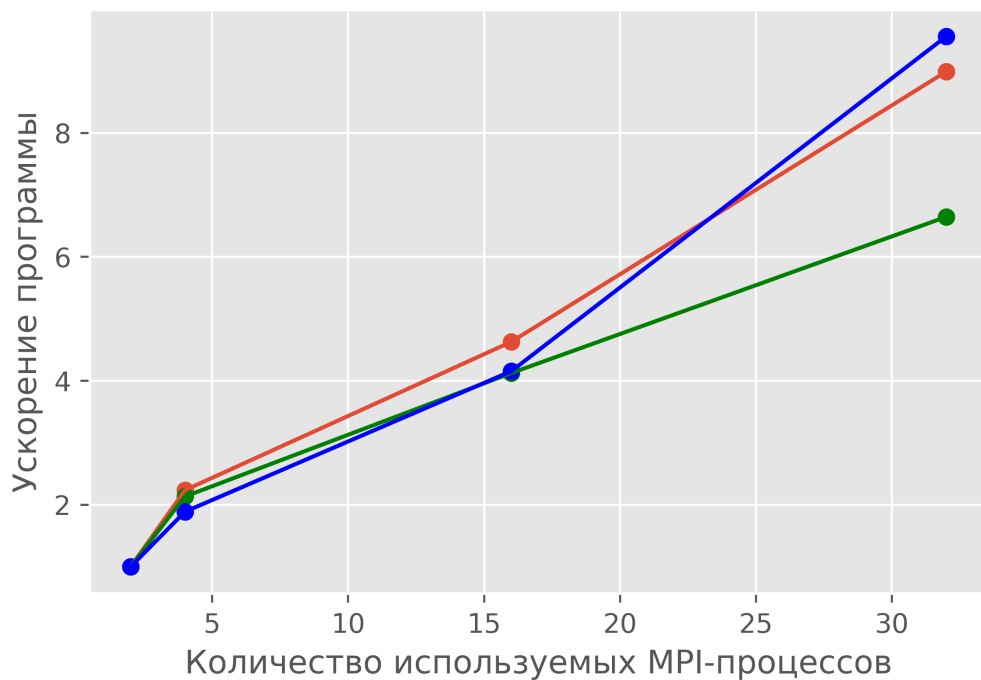


Рис. 6.4. Графики зависимости ускорения программы от числа используемых MPI-процессов **seed = 200**



Таблица 4. Таблица с результатами расчётов для **seed = 2000**

Точность $\varepsilon$	Число MPI-процессов	Время работы программы (с)	Ускорение	Ошибка
$3.0 \cdot 10^{-5}$	2	0.0541578240	1	0.0000067766
	4	0.0296152289	1.8287153606	0.0000067766
	16	0.0130133510	4.1617123829	0.0000067766
	32	0.0079458212	6.8158875762	0.0000067766
$5.0 \cdot 10^{-6}$	2	0.7897081950	1	0.0000025424
	4	0.4234589134	1.8648992145	0.0000025424
	16	0.2182825673	3.6178253021	0.0000025424
	32	0.0971342863	8.1300663759	0.0000025424
$5.0 \cdot 10^{-6}$	2	2.7588429691	1	0.0000009065
	4	1.6107378143	1.7127821453	0.0000009065
	16	0.8365108941	3.2980359114	0.0000009065
	32	0.3810515210	7.2400786168	0.0000009065

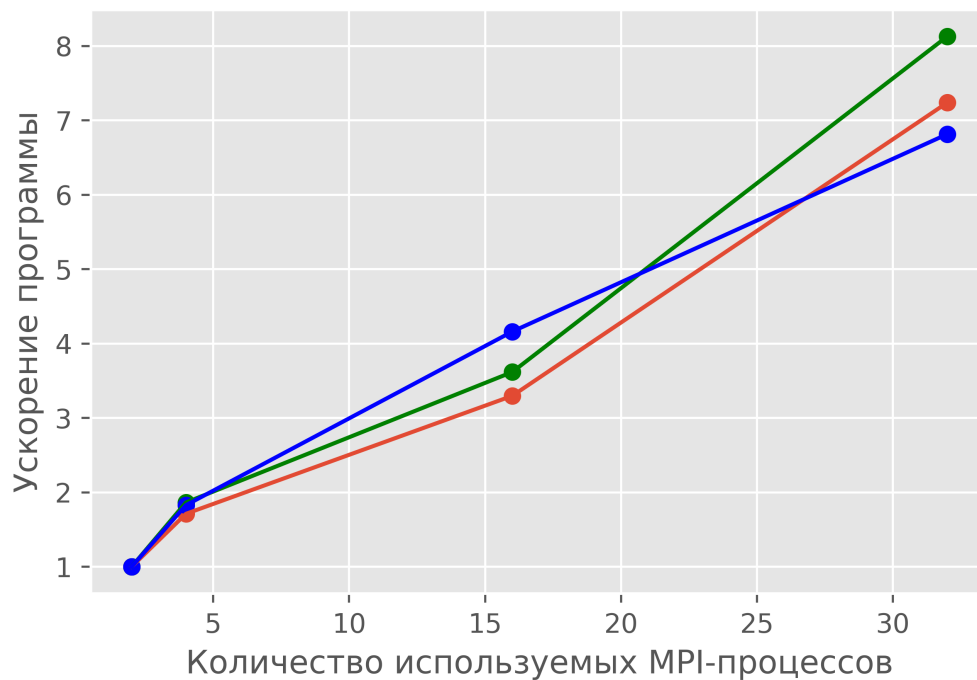


Рис. 6.5. Графики зависимости ускорения программы от числа используемых MPI-процессов **seed = 2000**

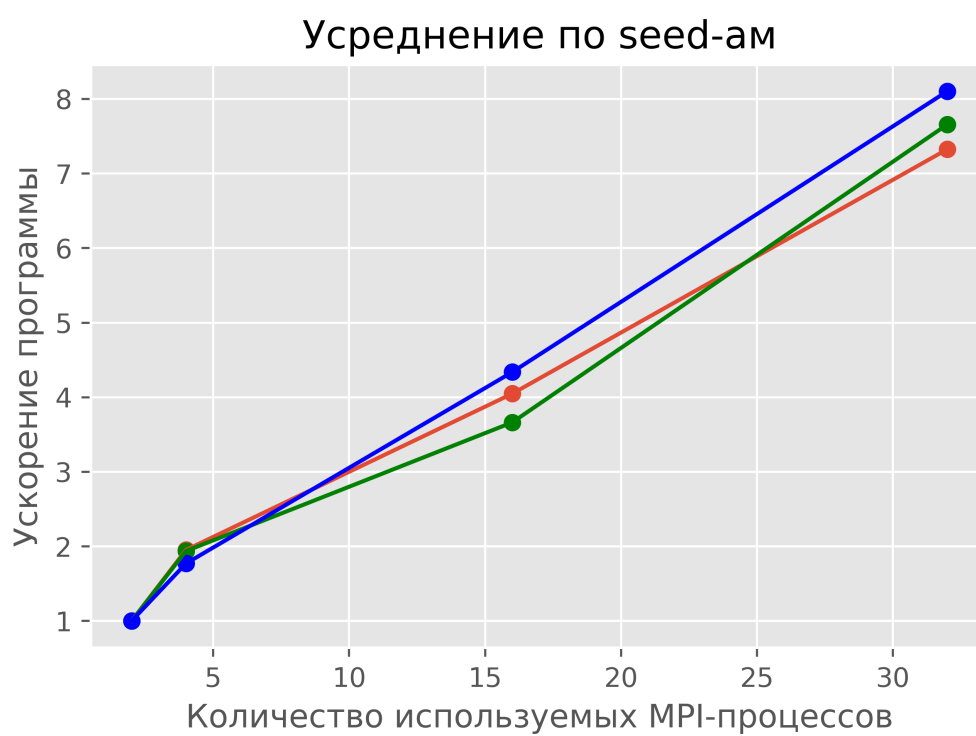


Рис. 6.6. Графики зависимости ускорения программы от числа используемых MPI-процессов со значениями усредненными по seed'ам